The Recursive Adjacency Decomposition Method

Mathieu Dutour Sikirić

January 16, 2010

I. Basic definitions

Polytopes, definition

- A polytope $P \subset \mathbb{R}^n$ is defined alternatively as:
  - The convex hull of a finite number of points $v^1, \dots, v^m$:

$$P = \{v \in \mathbb{R}^n \mid v = \sum_i \lambda_i v^i \text{ with } \lambda_i \geq 0 \text{ and } \sum \lambda_i = 1\}$$

  - The following set of solutions:

$$P = \{x \in \mathbb{R}^n \mid f^i(x) \geq b_i \text{ with } f_i \text{ linear}\}$$

    and P is bounded.
- The cube is defined alternatively as
  - The convex hull of the 2^n vertices

$$\{(x_1, \dots, x_n) \text{ with } x_i = \pm 1\}$$

  - The set of points $x \in \mathbb{R}^n$ satisfying to

$$x_i \leq 1 \text{ and } x_i \geq -1$$

Facets and vertices

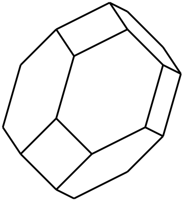
- A vertex of a polytope P is a point $v \in P$, which cannot be expressed as $v = \lambda v^1 + (1-\lambda)v^2$ with $0 < \lambda < 1$ and $v^i \in P$.
- A polytope is the convex hull of its vertices and this is the minimal set defining it.
- A facet of a polytope is an inequality $f(x) - b \geq 0$, which cannot be expressed as $f(x) - b = \lambda(f^1(x) - b_1) + (1-\lambda)(f^2(x) - b_2)$ with $f^i(x) - b_i \geq 0$ on P.
- A polytope is defined by its facet inequalities. and this is the minimal set of linear inequalities defining it.
- The dual-description problem is the problem of passing from one description to another.

Faces

- Given an inequality $f(x) \geq b$, which is valid on P, the face defined by $f(x) \geq b$ is

$$x \in P \text{ such that } f(x) = b$$

  and its dimension is the dimension of the smallest affine plane containing it.
- The dimension of faces of a n dimensional polytope P varies from 0 to $n-1$. A face of dimension 0 is a vertex, a face of dimension $n-1$ is a facet.
- Faces are defined by the set of vertices contained in them.
- The inclusion relation between faces defines a lattice.



Homogeneous coordinates and duality

- Linear functions are expressed in terms of scalar product.

$$f(x) = a_1 x_1 + \cdots + a_n x_n = \langle a, x \rangle$$

- A polyhedral cone is a cone defined by linear inequalities $f(x) \geq 0$. The vertices correspond to extreme ray.
- Formulas are easier for the polyhedral cones, all programs are designed for polyhedral cones and not for polytopes.
- But we can reduce polytope to polyhedral cones:
  - If $v \in \mathbb{R}^n$ is a vertex then we map it to a vector $v' = (1, v) \in \mathbb{R}^{n+1}$.
  - If $f(x) = \langle a, x \rangle \geq b$, we map it to a vector $a' = (-b, a)$.
  - The inequality $f(v) \geq b$ is then rewritten as $\langle v', a' \rangle \geq 0$.
- The two problems:
  1. given the vertices of P, find the facets,
  2. given the facets of P, find the vertices,

  are now expressed exactly identically:

  Find extreme rays of the cone $\langle a_i, x \rangle \geq 0$ with $1 \leq i \leq m$

II. What is generally easy

Linear algebra computations

- Suppose we have a n-dimensional polytope P and its list $\mathcal{LV}$ of vertices and we want to test if an affine inequality $f(x) \geq 0$ defines a facet.
  - We check if $f(v) \geq 0$ for all vertices $v \in \mathcal{LV}$
  - We compute the set of vertices $v \in \mathcal{LV}$ such that $f(v) = 0$. The rank of the defined space has to be $n - 1$.

  Similarly we can test if two facets are adjacent.
- Suppose we have a n-dimensional polytope P and its vertex-set $\mathcal{LV}$ and facet-set $\mathcal{LF}$.
  - We can compute all the face-set with rank computation only.
  - All question related to faces can be resolved.

Linear programming

- If $f(x)$, $f_i(x)$ are affine functions on $\mathbb{R}^n$, $b_i \in \mathbb{R}$, then the linear programming problem is:

$$\begin{aligned} \textbf{maximize} \quad & f(x) \\ \textbf{subject to} \quad & f_i(x) \geq b_i \end{aligned}$$

- Two main class of methods exist:
  - **The simplex method**: It goes from one vertex of the solution to another adjacent vertex until an optimal vertex is obained.
    NP in general, very good in practice.
  - **Interior point methods**: It takes an interior point and converges to a better and better vertex.
    With the primal dual method the method returns an interval, which can be made as small as possible.
    P in theory, relatively bad in practice.

  Generally we use simplex methods because they use exact arithmetic and for the kind of computation is usually not the limiting factor.

Computations related to linear programming

- Take $P = conv(v_1, \ldots, v_M)$ a polytope.
  - Testing if an element v belongs to the interior of P is lin.prog.
  - Testing if an element v belongs to P is lin.prog.
  - Determining the vertices amongst the v_i is lin.prog (M times).
  - Determining the adjacency $v_i - v_j$ amongst the v_i is lin.prog ($M(M-1)/2$ times).
- Take $P = \{x \in \mathbb{R}^n \ : \ f_i(x) \geq b_i \text{ for } 1 \leq i \leq N\}$.
  - Testing $P = \emptyset$ is lin.prog.
  - Computing the dimension of P is lin.prog.
  - Determining facet defining inequalities is lin.prog.
  - Finding one vertex is lin.prog.
- In principle we can obtain all the facets from such linear combinations but we will see faster methods.
- Linear programming is ok, when not used too much. If that is the case, then it is better to use linear algebra method.

III. The dual description problem

Computing dual description

- The dual description problem is important to many many computations:
  - It allows to test membership questions easily.
  - It allows to get the full face-set if needed.
- In high dimension the problem becomes difficult:
  - The number of vertices, facets grows very fast.
  - Even if the number is small, it can be difficult to compute.
- Some known programs exist (`cdd`, `lrs`, `ppl`, `pd`, `porta`, `qhull`, etc.), their efficiency varies widely and sometimes they take too much time.
- In many cases the polytope considered have a "big" symmetry group and the orbits of facets is the really needed information.
- We will expose some techniques for dealing with this problem.

Limitations of the hope

- If the quotient $\frac{\#facets}{|G|}$ is really too large then the problem becomes impossible.
- Combinatorial explosion is the driving phenomenon. Using symmetry has only limited efficiency.

polytope	dimension	$\|V\|$	$\|G\|$	# orbits	# facets
CUT_4	6	8	1152	1	16
CUT_5	10	16	1920	2	56
CUT_6	15	32	23040	3	368
CUT_7	21	64	322560	11	116764
CUT_8	28	128	5160960	147?	
CUT_9	36	256	185794560	$\geq 1.10^6$	

- In practice, the method explained here allows to gain one more step.

Program comparisons

We consider a polytope defined by inequalities $\mathcal{LF}$ for which we want its vertices.

- lrs: it iterates over all admissible basis in the simplex algorithm of linear programming
  - It is a tree search, no memory limitation.
  - Some repetition can occur in the output.
  - Ideal if the polytope has a lot of vertices.
- cdd: it adds inequalities one after the other and maintain the dual description through
  - All vertices and facets are stored, memory limited.
  - Good performance if the polytope has degenerate vertices.
- pd: We have a partial list of vertices, we compute the facets with lrs. If it does not coincide with $\mathcal{LF}$ then we can generate a missed vertex by linear programming.
  - It is a recommended method if there is less vertices than facets.
  - Bad performance for general polytopes.
- So, in general, choosing the right method is really difficult.

The adjacency decomposition method

`Input`: The vertex-set of a polytope P and a group G acting on P.
`Output`: $\mathcal{O}$, the orbits of facets of P.

- compute some initial facet F (by linear programming) and insert the corresponding orbit into $\mathcal{O}$ as `undone`.
- For every `undone` orbit O of facet:
  - Take a representative F of O.
  - Find the ridges contained in F, i.e. the facets of the facet F (this is a dual description computation).
  - For every ridge R, find the corresponding adjacent facet F' such that $R = F \cap F'$.
  - For every adjacent facet found test if the corresponding orbit is already present in $\mathcal{O}$. If no insert it as `undone`.
  - Mark the orbit O as `done`.
- Terminate when all orbits are `done`.

History of the method

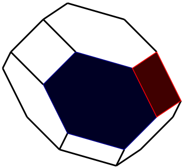
The Adjacency decomposition method is perhaps the most natural method for computing orbits of facets.
The method was reinvented many times

- "Voronoi algorithm" by Voronoi (1908) (perfect domains)
- "Algorithme de l'explorateur" by Jaquet (1993) (facets of perfect domains)
- "Adjacency decomposition method" by Christof and Reinelt (1996) (Linear Ordering Polytope, Traveling Salesman Problem, Cut Polytope)
- "Subpolytope algorithm" by Deza et al. (2001) (Metric Polytope)

General feature of the algorithm

A "forest fire":

- The algorithm starts by computing the orbits of lowest incidence, which are the one for which the dual description is easiest to be done.
- Sometimes it seems that no end is in sight, we get a lower bound on the number of orbits.
- At the end, only the orbits of highest incidence remains.
- In most cases, the orbits of highest incidence do not yield new orbits but in a few cases, this happened.



Balinski theorem

The skeleton of a polytope is the graph formed by its facets with two vertices adjacent if and only if the facets are adjacent.

- Balinski theorem The skeleton of a n-dimensional polytope is n-connected, i.e. the removal of any set of $n-1$ vertices leaves it connected.
- So, if the number of facets in remaining orbits is at most $n-1$, then we know that no more orbits is to be discovered.

Scope of application:

- the criterion is usually not applicable to the polytopes of combinatorial optimization, i.e. the orbits of facets of such polytopes are usually relatively big.
- For the polytopes arising in geometry of numbers, it is sometimes applicable.
- very cheap to test, huge benefits if applicable.

The recursive adjacency method

In all cases considered so far, the orbits of maximum incidence also have the highest symmetry and are the most difficult to compute.

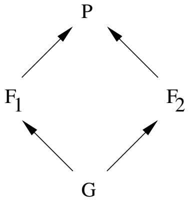
- The computation of adjacent facets is a dual-description computation.
- So, the idea is to apply the Adjacency Decomposition method to those orbits as well.
- Based on informations on the symmetry group and on the incidence, we decide if we should launch the adjacency method.

Issues:

- The number of cases to consider can grow dramatically.
- If one takes the stabilizer of a face, then the size of the groups involved may be quite small to be efficient.

Banking methods

- When one applies the Recursive Adjacency decomposition method, one needs to compute the dual description of faces.
- F_1 and F_2 are two facets of P to which we apply the Adjacency Decomposition Method.
  G is a common facet of F_1 and F_2.
  The dual description of G is computed twice:



- The idea is to store the dual description of faces in a bank and when a dual description is needed to see if it has been already done.

Possible improvements

There are still some possible ways to improve the programs:

- Better choice of heuristics.
  - How to choose the dual description program? So far, we use only `lrs`.
  - When to respawn a new adjacency computation?
  - When to save the dual description in the bank?
  - When to use stabilizer of a face or its inner symmetry group?
- Sometimes the heuristics make a choice that leads to a too long computation. It would be good if this could be dealt with.
- Use parallel processing with `ParGAP`.

IV. Symmetry questions

Permutation groups

- Polytopes of interest have usually less than 1000 vertices $v_1, \dots, v_N$, their symmetry group can be represented as a permutation of their vertex-set.
- The first benefit is that permutation group algorithms have been well studied for a long time and have good implementation in `GAP`.
  - A. Seress, *Permutation group algorithms*, Cambridge University Press, 2003.
  - D.F. Holt, B. Eick and E.A. O'Brien, *Handbook of computational group theory*, Chapman & Hall/CRC, 2005.
- The second benefit is that a facet of a polytope thus corresponds to a subset of $\{1, \dots, N\}$ and that permutation group acting on sets have a very good implementation in `GAP`.
- In some extreme cases (millions of vertices) permutation groups might not work as well and other methods have to be used.

Symmetry questions

Usually, most of the computational time is spent in symmetry computations.

- We always need two operations:
  - Isomorphism tests between two objects.
  - Computation of the stabilizer or automorphism group of an object.
- There are three different contexts:
  - Identifying orbits when the full orbit has been generated.
  - Given a polytope P and a group G acting on P, test if two faces are equivalent under G.
  - Test if two polytopes are isomorphic.

Full orbit

- Eventually, the Recursive Adjacency Decomposition Method will call `lrs`, `cdd`, etc for generating the full dual-description.
- Hence, the full orbits of facets will be generated,
- The idea is then to code those orbits by 0/1-vectors and to identify the full orbits.
- This is potentially memory-limited but extremely efficient in `C++`.
- In 2G of RAM we can handle only 20 million facets. This is sometimes a problem dealt with by an additional respawn of adjacency method.

In the Adjacency decomposition iteration

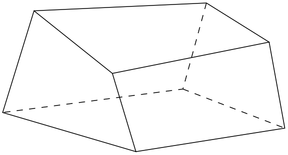
We have a fixed group G of a polytope P and we want to test if two faces F_1 and F_2 are equivalent under G.

- We represent G as permutation group on the set of vertices $(v_i)_{1\leq i\leq N}$ of P and the faces by their incidence, i.e. subsets of $\{1, \dots, N\}$.
- Then, we use two following functions in GAP
  - Stabilizer(G, S1, OnSets);
  - RepresentativeAction(G, S1, S2, OnSets);

  The important fact is that the action OnSets is extremely efficient and uses backtrack search, i.e. in practice we never build the full orbit.
- The main reason why our program is working is because GAP has efficient implementation of those functions.

Combinatorial symmetry group

- The combinatorial symmetry group of a polytope P, which permutes the faces of P, while preserving the inclusion relation.
- This is the most natural group for this problem
- Since every face is described by its vertices, this group can be represented as a permutation group on m elements if P is the convex hull of m vertices $v^1, \dots, v^m$.
- It can be proved that we need "only" the facet to compute this group.
- But knowing the facets is the goal itself, so we have to settle to smaller groups



Symmetry group of polytopes

We take a rank n family of vector $(v_i)_{1\leq i\leq N}$ in $\mathbb{R}^n$.

- An automorphism of this vector family is a matrix A such that

$$v_i A = v_{\sigma(i)} \text{ for some } \sigma \in Sym(N)$$

  We want to compute the group of automorphism of the vector family.

- Define the form

$$Q = \sum_{i=1}^{N} {}^t v_i v_i$$

- Define the edge colored graph on N vertices with edge color

$$c_{ij} = v_i Q^{-1}\, {}^t v_j$$

- The automorphism group of the edge colored graph corresponds to the automorphism group of the vector family.

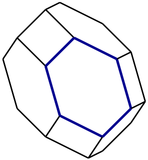
- The automorphism group of the edge colored graph is computed with nauty and a reduction to a vertex colored graph. If G has n vertices and k colors, then we have the following reductions:
  - Line graph: $\frac{n(n-1)}{2}$ vertices.
  - Every color is a graph: nk vertices.
  - Every bit of a color is a graph: $n \log(k)$ vertices.
  - Another construction: $n\sqrt{\log(k)}$ vertices.
- PROBLEM: The projective automorphisms of a polyhedral cone are the matrices A such that

$$v_i A = \alpha_i v_{\sigma(i)} \text{ with } \alpha_i > 0 \text{ and } \sigma \in Sym(N)$$

i.e. A permutes the extreme rays.
We do not know how to compute this group efficiently.

Symmetries and orbit mapping

- The symmetry group of the face might be larger than its stabilizer under the bigger group.



  - The stabilizer of the face has order 6
  - The symmetry group of the face has order 12.
- Suppose that we have a set of orbits for the big symmetry group G

$$\mathcal{F} = x_1 G \cup \cdots \cup x_n G$$

  we want to represent $\mathcal{F}$ as list of orbits for a subgroup H of G.
- For every x_i do a double coset decomposition

$$G = G_{x_i} g_1 H \cup \ldots \cup G_{x_i} g_p H$$

  with G_{x_i} the stabilizer of x_i in G.
- So, $x_i G = \cup_j x_i g_j H$

V. Case Studies

Perfect domain $Dom(E_8)$

- Context: The Voronoi algorithm for computing perfect forms in dimension n needs to find the facets of their perfect domains.
- The perfect domain $\mathrm{Dom}(E_8)$ has 120 extreme rays and is of dimension 36 symmetry group has size 348364800.
- There are 25075566937584 facets in 83092 orbits.
- 4 orbits required a secondary application of the ADM.
- The orbit made of facets of incidence 75 have a stabilizer of size 23040 but a symmetry group of size 737280, therefore allowing us to finish the computation.
- Total running time with ons and offs was 15 months.

Contact polytope of O_{23}

- Context: The determination of overlattice of O_{23} of minimum 3 requires the computation of vertices of the contact polytope of O_{23}.
- The polytope $Contact(O_{23})$ has 4600 facets, dimension 23 and the symmetry group $\mathbb{Z}_2 \times Co_1$.
- There are 15615584979368414 vertices in 269 orbits.
  - One vertex correspond to a 22-dimensionl simplicial polytope of 44 vertices with a group transitive on simplices.
  - HS, M_{22}, M_{23} appear as stabilizer of vertices.
  - One orbit is incident to 275 facets and has group McL.
  - One orbit of simplices is incident only to the above orbit.
- Main computational difficulty is in checking if two vertices are equivalent.
- Total running time is two days.

VI. Related methods

The incidence technique

The incidence technique is the logical competitor of the Adjacency Decomposition Method.

- Suppose that the vertex set $\mathcal{E}$ of P is partitionned into orbits $\{O_1, \dots, O_p\}$ of representative v_i.
- For every $1 \leq i \leq p$, consider the space

$$P_i^* = \{f \in (\mathbb{R}^n)^* \mid f(v) \geq 0 \text{ for } v \in \mathcal{E} \text{ and } f(v_i) = 0\}$$

  Every facet of P is equivalent to a facet in P_i^* for some i.
- The description of P_i^* may be redundant, so elimination of redundancy by linear programming is necessary.
- The incidence method admits extensions to edges of P, 2-dimensional faces of P, etc.

The cascade algorithm

The Cascade algorithm (a reincarnation of Fourier-Motzkin) has been introduced by Jaquet (1992):

- If P is a polytope of dimension n with m vertices, then it is the projection of a simplex of dimension $m-1$.
- If P' is a polytope in $\mathbb{R}^q$, f a projection on an hyperplane of $\mathbb{R}^q$, then the facets of $f(P')$ are:
  - Projections of facets of P'.
  - Projections of intersection of adjacent facets of P'.
- Using `GAP`, we can compute the orbits of facets of the projection $f(P')$ from the orbits of facets of the polytope P'.
- This yields an algorithm for enumerating facets of P.
- The problem is that the intermediate polytopes have a much smaller symmetry group than the original polytope.

Face-lattice under symmetry

The face-lattice of a polytope is usually "fat":

- The number of faces of intermediate dimension is much larger than the number of vertices and facets.

There is an efficient algorithm for enumerating the faces under symmetry:

- We first compute the facets of the polytope.
- We represent faces by the list of incident vertices and the action OnSets.
- For every face F of dimension k, we use the facets to find the faces of dimension $k+1$ to which F belongs.
- We then reduce by isomorphism.

Flag system under symmetry

- The number of flags is much larger than the number of faces.
- But there is an efficient algorithm for enumerating orbits of flags under symmetry.
- The idea is to extend flags $(F_0, \ldots, F_k)$ to flags $(F_0, \ldots, F_{k+1})$ with $\dim F_i = i$.
- The only trick is the isomorphism test:
  - Take two flags $f = (F_0, \ldots, F_k)$ and $f' = (F'_0, \ldots, F'_k)$
  - Check isomorphism of F_0 and F'_0 under G with OnSets. If not-isomorphic leave.
  - If $F'_0 = F_0.g$ then replace f by $f.g$.
  - Replace G by the stabilizer of F'_0.
  - Consider faces of dimension $1, \ldots, n$.

Availability

The software polyhedral is available from my web page

`http://www.liga.ens.fr/~dutour/polyhedral/`

Other features:

- The system works, optionally, by saving to disk:
  - This works by guaranteeing atomicity of operations.
  - This is useful in case of power failure, no loss of work.
  - If some problem show up, we can rerun from where we were with other settings.
- Written in `GAP`, `perl`, `C++` using many people's other programs (`nauty`, `cdd`, `lrs`).
- Examples, but no manual yet.

THANK

YOU